Simultaneous Gate and Subthreshold Leakage Reduction Techniques

Ashley Brinker, Karen Joseph,
Mehdi Kabir
ECE 6332 – Fall 2010
University of Virginia
<aab6b,kjj8v,mk4kg>@virginia.edu

ABSTRACT

In this paper we will discuss the effects of gate and subthreshold leakage reduction techniques on arbitrary circuits. According to the present scaling trends, power will become a limiting factor in future VLSI technologies. With the scaling down of transistors leakage power becomes a significant portion (more than 40%) of the total power consumed in modern chips. Many leakage reduction techniques ignore gate leakage although it makes up a large portion of the total leakage. In order to optimize the leakage power simultaneous gate and subthreshold leakage reductions are needed. In our analysis we used both input vector control and MTCMOS to observe the effects on leakage reduction. Using our heuristic for finding the optimal input vector of any arbitrary circuit we saw up to 28% reductions in total leakage current. We then used MTCMOS to find an optimal V_T for a given process size which maximized the current-delay product. Finally we combined the input vector control and the MTCMOS for further leakage reductions seeing up to 45% reduction in gate current.

1. INTRODUCTION

Over the past few decades, CMOS devices have been scaled down to achieve higher performance and lower power consumption. Transistor delay has decreased by more than 30% per technology generation, doubling a microprocessor's performance every two years. Accordingly, supply voltage is scaled down causing the threshold voltage to also be scaled down in order to maintain a high drive current. However, scaling threshold voltage increases subthreshold leakage current. Ideally, when a transistor is off, there should be no conduction between the drain and the source. However, conditions such as weak-inversion, carrier concentration gradient, and thermal effects cause a subthreshold leakage current [1].

Increasing channel conductivity increases performance and reduces subthreshold leakage. However, this causes electron tunneling through the gate, creating a gate leakage current. As the process size of a transistor is scaled down from 130nm to 32nm technology, the gate leakage becomes a larger portion of the total leakage starting at 19% for 130nm and rising to 38% for 32nm as shown in the Figure 1.

Subthreshold reduction techniques can cause an increase in gate leakage and as gate leakage is a significant portion of the total leakage, in many cases, better leakage reduction can be achieved if gate leakage current is optimized first.

Current scaling trends in CMOS technologies indicate that leakage power is more than 40% of the total power consumed, with gate leakage playing a significant role in the total leakage. This paper looked at using input vector control and multiple-threshold CMOS (MTCMOS) as methods of simultaneous reduction of both subthreshold and gate leakage and its effect on performance (delay).

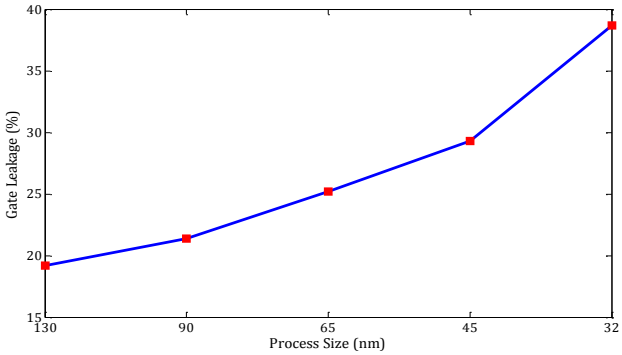


Figure 1. Gate leakage current as a function of process size.

2. APPROACH

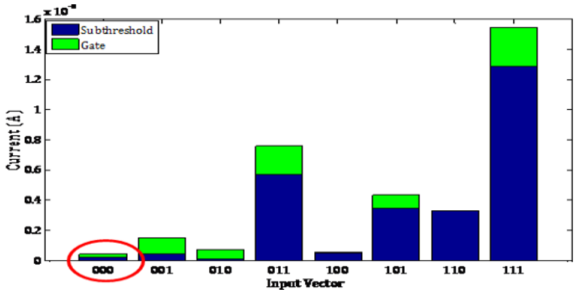
OCEAN simulations were done using the Predictive Technology Models for 32nm, 45nm, 65nm, 90nm, and 130nm technologies. The supply voltage was kept at a constant 1.0V and all the transistors were sized equally. To analyze the effect of input vector control and MTCMOS, these techniques were first implemented on simple 3-input NAND or 3-input XOR gates and compared the leakage reduction across different technologies. For MTCMOS, changes in threshold voltage and its tradeoff with delay were also analyzed. These techniques were then applied to a 16-bit ripple carry adder to determine an optimal leakage reduction design.

3. INPUT VECTOR CONTROL

One effective method for reducing both gate and leakage control is input vector control. This technique takes advantage of the fact that for a given logical circuit, the total leakage through the circuit is depends on the input data of the circuit. In Figure 2(a) we see the leakage breakdown between gate and subthreshold leakage for all the inputs of a 3-input NAND gate. By forcing the inputs to adopt the lowest leakage configuration, the leakage can be reduced significantly over the average leakage current of uniformly distributed inputs.

It is also important to note that process size also has an impact on the optimal input vector. As shown in Figures 2(a-b), the optimal input vector changes for the 3-input NAND gate from *<000>* to *<100>* as we scale down the process from 130nm to 32nm. As was shown above, gate leakage plays a more prominent role in scaled down devices and therefore the optimal input vectors tend to be ones with the least amount of gate leakage.

(a)



(b)

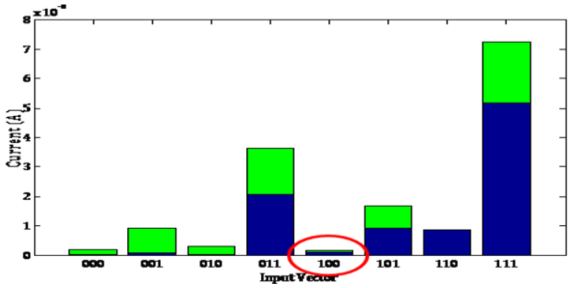


Figure 2. Total leakage breakdown in 3-input NAND gate in (a) 130-nm process size (b) 32-nm process size. The optimal input is circled for each process size.

3.1 Minimum Leakage Vector

In order to achieve the largest current savings, we need to find the input vector, called the minimum leakage vector (MLV), which yields the lowest leakage through the circuit. Finding the MLV for an arbitrary circuit is an NP-complete problem [2].

There have been many different approaches to solve this problem. For small circuits, an exhaustive search is the simplest method and one that guarantees the correct MLV. However, for circuits with large number of inputs and/or logic depth, an exhaustive search can become intractable. In such cases, random genetic algorithms and Monte Carlo methods can be used to sample a smaller set of input vectors and yield the MLV [3].

Another set of algorithms falls under the category of a greedy search. A greedy search creates a cost function that evaluates the effectiveness of setting a small subset of the input vector on the total leakage of the circuit. In each iteration of the algorithm, the subset of input vectors which produces the least leakage through the circuit is assigned to its optimal vectors. The process is repeated until all the input vectors have been assigned a value. In [4], the author discusses effective cost functions which can be used towards the MLV problem. However, due to its myopic nature, the greedy algorithm is susceptible to be "trapped" in a local minimum of the input vector space when there exists a better global minimum. In order to mitigate this problem, the idea of node controllability is introduced to solve a similar greedy search problem (gate replacement) in [2]. In our work, we decided to use this principle towards the MLV problem to increase the accuracy of our search.

3.1.1 Node Controllability

Node controllability tries to find the minimum number of inputs required to force a node in a circuit to either a logic '1' or '0'. The node controllability of the circuit in Figure 3 is shown in the Table 1(a). By creating a node controllability list, we anticipate possible conflicts that can arise by optimizing the inputs of one gate on the other gates of the circuit. In this way, the greedy algorithm can take into some of the global effects of a certain input vector.

We can generate the list of conflicting gates by first finding the optimal vector for a gate based on the constraints of the node controllability list. For the example circuit, a 2-input NAND gate has optimal inputs when its second input is '0'. Therefore, to find the optimal vector for gate C4, we require that N7 node equal '0' which we can retrieve from the node controllability list. In cases of multiple constraints, the input vectors can be ANDed together to give the optimal input for that gate. The optimal gates inputs are shown in Table 1(b).

Once we have the optimal vector list for all the gates, we can see that some gates have complementary relationships (gates C1 and C4), while others conflict (P3 input for C1 and C2 gates). This can be taken into account when determining the cost function.

3.1.2 Cost Function

The cost function for the greedy search is based on the analysis from [2] and [4]. The cost function uses a penalty function for each type of gate, which awards circuits that complement the most amounts of gates and punishes those that create the most conflict. The penalty function is usually a weighted average of unwanted input states minus the weighted average of the wanted states. For the example circuit in Figure 3, the penalty function was:

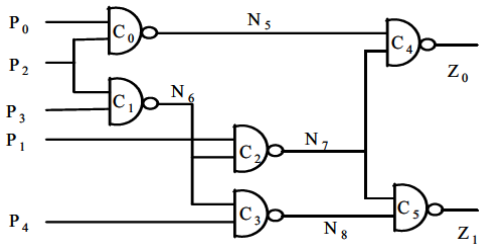


Figure 3. Example circuit (benchmark circuit c17 from ISCAS).

Table 1. (a) Node controllability list for the circuit in Figure 3. Shows the list of <*P0…P4*> inputs along with the minimum number inputs required (note: 'x' means "don't care").

(b) Shows the optimal vector given the constraints imposed by the node controllability list.

Node	CC0	CC1
N5	2: (1x1xx)	1: (0xxxx)
N6	2: (xx11x)	1: (xxx0x)
N7	2: (x1x0x)	1: (x0xxx)
N8	2: (xxx01)	1: (xxxx0)
Z0	2: (00xxx)	2: (1x1xx)
Z1	2: (x0xx0)	2: (xxx01)

Gate	Optimal Vector
C0	(P2=0)→xx0xx
C1	(P3=0)→xx0x
C2	(N6=0)→xx11x
C3	(P4=0)→xxxx0
C4	(N7=0)→x1x0x
C5	(N8=0)→xxx01

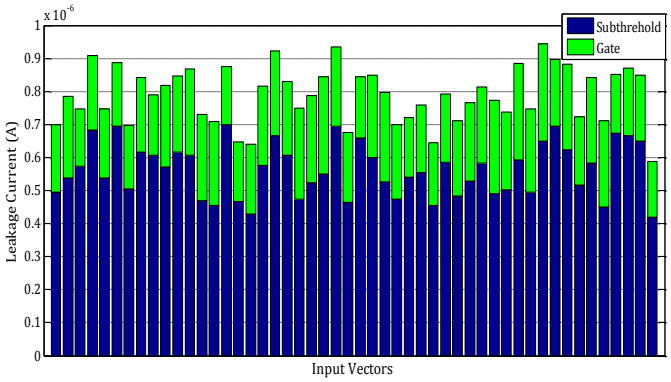


Figure 4. Leakage through 16-bit FA for 50 random input vectors. The MLV of the heuristic is the final data point.

$$PF(NAND2) = 0.5x(L_{01}+L_{11})-0.5x(L_{00}+L_{10}),$$

where the L represents the leakage for the particular input vector. The weights can be adjusted depending the on the severity of the leakage and also by the type of gate. The penalty function is only applied when a gate is not in its optimal vector (i.e. conditions in Table 1(b)). Finally the cost function for a gate can be evaluated as:

$$Cost(Gate_i) = \Sigma\, PF[Conflicting\ Gates\ (Gate_i)] - \Sigma\, PF[Complement\ Gates\ (Gate_i)] - PF\ (Gate_i)$$

3.1.3 MLV Algorithm

The greedy algorithm to find the MLV can be summarized as follows:

1. Generate the node controllability list for all the nodes in the circuit.
2. Generate a list for each gate find the optimal vector under the constraint of the node controllability list.
3. From the optimal vector create a list of conflicting and complementary gates for each gate.
4. For first iteration, put all gates in a selection list. Also set the MLV input vector *to* <*xxxxx*…..>.
5. While selection list is not empty
    a. Compute the cost function of each gate in list.
    b. Choose the gate with minimum cost function and adjust the MLV to the optimal vector of that gate.
    c. Remove the gate and its conflicting and complementary gates from the selection list.
    d. Update the conflicting and complementary gates of the remaining gates in the selection list.
6. For each unassigned input vectors:
    a. Compute the total cost of the circuit if the input is set to '0' and then do the same for '1'.
    b. Assign the vector to the input with the minimum cost.

This algorithm complexity is on the order of $O(n^2)$ where n is the number of gates in the circuit. We tested the heuristic on a 16-bit full adder using a MATLAB script. We compared the MLV generated by the algorithm to a set of 50 random input vectors and we concluded that it is the MLV with 98% confidence level (See Figure 4).

4. MTCMOS

MTCMOS is another common leakage reduction technique. As its name indicates, it utilizes transistors with different threshold voltages (V_T) to optimize power and delay. MTCMOS was used to reduce leakage current without having significant penalties on delay. Header and footer sleep transistors with high V_T were added to the circuit. When the sleep signal is on, the header and footer transistors turn off, cutting the circuit's path to V_{DD} or ground. Hence, leakage current would be determined by the high V_T transistors. When the circuit is in active mode (sleep signal is off), the delay would be determined by the low V_T transistors. To analyze the effects of the header and footer transistors on a circuit, we implemented MTCMOS for a simple 3-input NAND gate as shown in the figure below. We analyzed its effect on subthreshold leakage, gate leakage, and delay for different high V_T values (from 0.4V to 0.95V).

For the simulations, V_{DD} was kept constant at 1.0V and the input vector was forced at *<111>*. To measure delay, the input transition used was *<000>* to *<111>*. The low V_T devices were kept at a constant V_T of around 0.4V while the high V_T header and footer sleep transistors were varied from 0.4V to 0.95V in 0.05V intervals. The simulations were run for 32nm, 45nm, 65nm, 90nm, and 130nm technologies. As expected, the current decreased exponentially until a certain V_T and then saturated. The exponential curve is due to the exponential relationship between V_T and the leakage current described by the leakage equation. The saturation as V_T increases is possibly due to the gate-induced drain leakage (GIDL). This value is usually small and insignificant compared to the subthreshold leakage. However, as V_T increases, GIDL becomes a larger component of the subthreshold conduction and after a certain point, GIDL dominates. Since the current caused by GIDL is (nearly) constant for constant V_{DS}, the current reaches saturation as V_T increases.

For this input vector, subthreshold current is 100 times larger and hence dominates the total leakage current of the circuit. Figure 5 below shows the total leakage current of the circuit across different high V_T for different size technologies.

As expected, increasing V_T decreased current with a tradeoff in delay. To assess this tradeoff in delay, different metrics were analyzed. Different metrics raised the delay to different powers, hence weighing it differently. The minimum of the product curves gives an optimal V_T for that sized technology. For that metric, the current and delay are optimized at that V_T value. Figure 6 shows the trend in optimal V_T across different technologies for the different metrics.

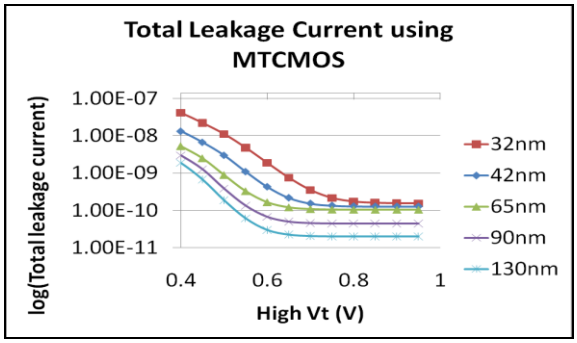


Figure 5. Total Leakage Current of MTCMOS NAND gate

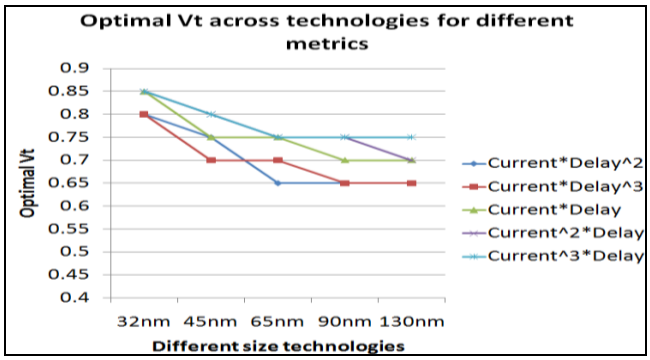


Figure 6. Optimal V_T across different size technologies for different metrics.

5. RESULTS & DISCUSSION

Using the results the input vector and MTCMOS, we applied it to a 16-bit full adder circuit. First, the heuristic was used to find the MLV for the adder circuit across different process size. We then compared that leakage with a set of 50 random input vectors and found that the MLV reduced leakage by 17%-28% depending on the process size. The reduction increased with scaling because of the MLV was effective in suppressing the gate leakage in conjunction with the subthreshold leakage. Just like the 3-input NAND gate, the MLV changed as we scaled from 130nm to 45nm.

We then attempted to reduce the leakage further by introducing MTCMOS gates. After the introduction of the MLV, we noticed that a large portion of the leakage currents were concentrated around few gates. These gates were typically ones that were forced into their worst case input vector by the heuristic algorithm. Since a large part of the leakage of these gates was due to subthreshold current, replacing them with an MTCMOS gate was a natural solution for leakage reduction (See Figure 7). We chose the optimal threshold voltage based on the Current-Delay^2 metric minimize the delay penalty. With the added MTCMOS, we were able to reduce the leakage by up to 45% compared to an unmodified random input vector full adder circuit (See Table 2 for full results) with a small delay penalty.

Our method of gate replacement is not a general solution for an arbitrary circuit. This is because by adding an MTCMOS we introduce a virtual ground and V_{DD} which can cause the outputs to float. Depending on the output voltage, this can introduce errors in the rest of the circuit. One way to mitigate this problem is to use high-V_T transistors for all the components of the gate instead of using it in header and footers only. However, this comes at the cost of increased delay during active operation time and increase of leakage due to the removal of stack of the MTCMOS sleep transistors.

6. CONCLUSIONS

In this paper we showed the importance of gate leakage when considering leakage reduction techniques. By using input vector control and MTCMOS we were able to simultaneously reduce both sources of leakage by up to 45% in a full adder circuit. In addition, we developed a heuristic to determine the MLV for an arbitrary logic circuit and showed a proof of concept on the 16-bit full adder. Further work can be done to use this heuristic and apply it to other benchmark to gauge its accuracy. Other avenues of investigation would include coming up with a systematic approach to gate replacement which would maximize the total leakage reductions.

7. REFERENCES

[1] Roy, K., Mukhopadhyay, S., Mahmoodi, H. Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits. *Proc. IEEE, Vol. 91, No. 2.*

[2] Yuan, L., Qu, G. A Combined Gate Replacement and Input Vector Control Approach for Leakage Current Reduction. *IEEE Trans. VLSI Systems 2006, 14(2).*

[3] Johnson, M., Somasekhar, D. and Roy, K., "Models and Algorithms for Bounds in CMOS Circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems,* Vol. 18, No. 6, June 1999, pp. 714-725.

[4] A. Abdollahi *et al*. Leakage Current Reduction in CMOS VLSI Circuits by Input Vector Control. *IEEE Trans. VLSI Systems 2004, 12(2).*

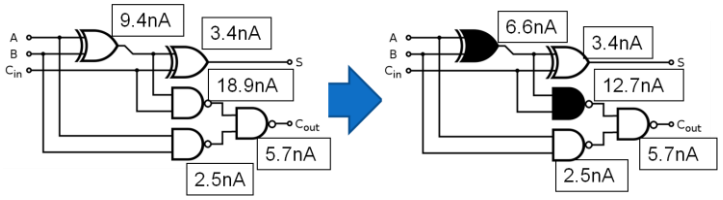


Figure 7. Gate replacement with MTCMOS in a bitslice of 16-bit full adder.

Table 2. Leakage Reduction by: MLV (Step #1) and MLV+MTCMOS (Step #2).

Process Size	MLV	Step #1	Step #2	Delay Penalty
130 nm	A = 0000…0000 B = 0000…0000 Cin0 = 0	17.6%	28.9%	3.2%
90 nm	A = 0000…0000 B = 0000…0000 Cin0 = 0	19.4%	30.4%	3.4%
65 nm	A = 0000…0000 B = 0000…0000 Cin0 = 0	21.5%	33.9%	3.4%
45 nm	A = 0000…0000 B = 1010…1010 Cin0 = 1	25.3%	41.8%	4.7%
32 nm	A = 0000…0000 B = 1010…1010 Cin0 = 1	28.3%	45.2%	5.6%